# Esame UFS04 - Algoritmi di Machine Learning

**OCR Carta d’identità**

# *Esercitazione svolta da Formenti Roberto, Romanazzi Matteo, Segalini Michele, Toninelli Luca*

**Indice:**

[Introduzione](#_vqp82o5thg0u) 2

[Setup](#_46eht2mrlejd) 2

[OCR & KNN](#_f46dhshe0hci) 3

[Flask](#_uuu1sdjb9m1h) 4

[Implementazione Web](#_g4h41ha1i7lv) 5

[Problemi riscontrati](#_izboxi72hd86) 5

[Conclusione](#_cyrvl6tm8tws) 5

## 

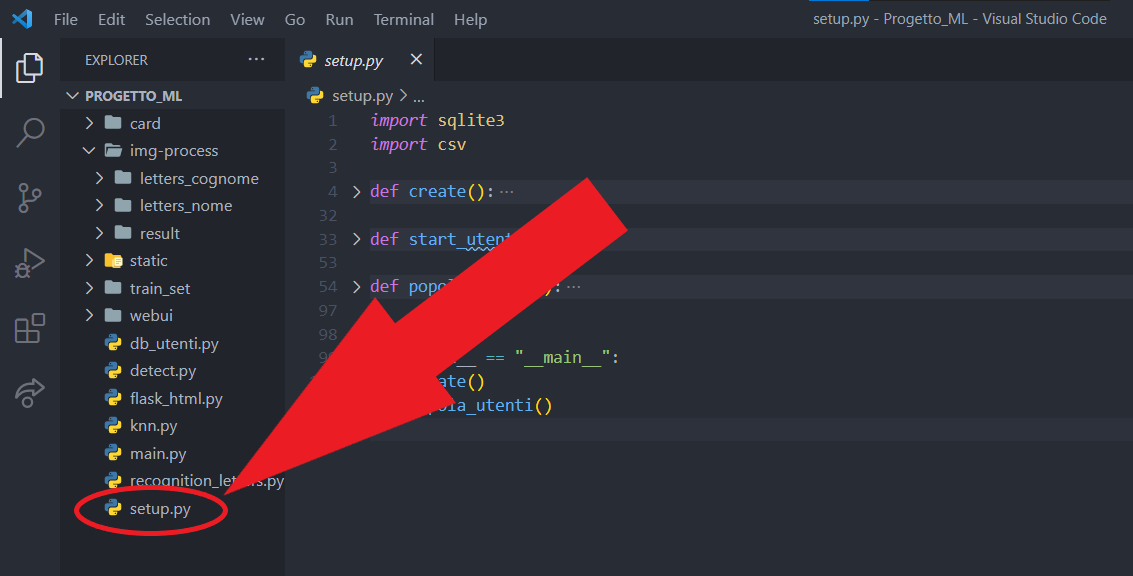
## 

## **Introduzione**

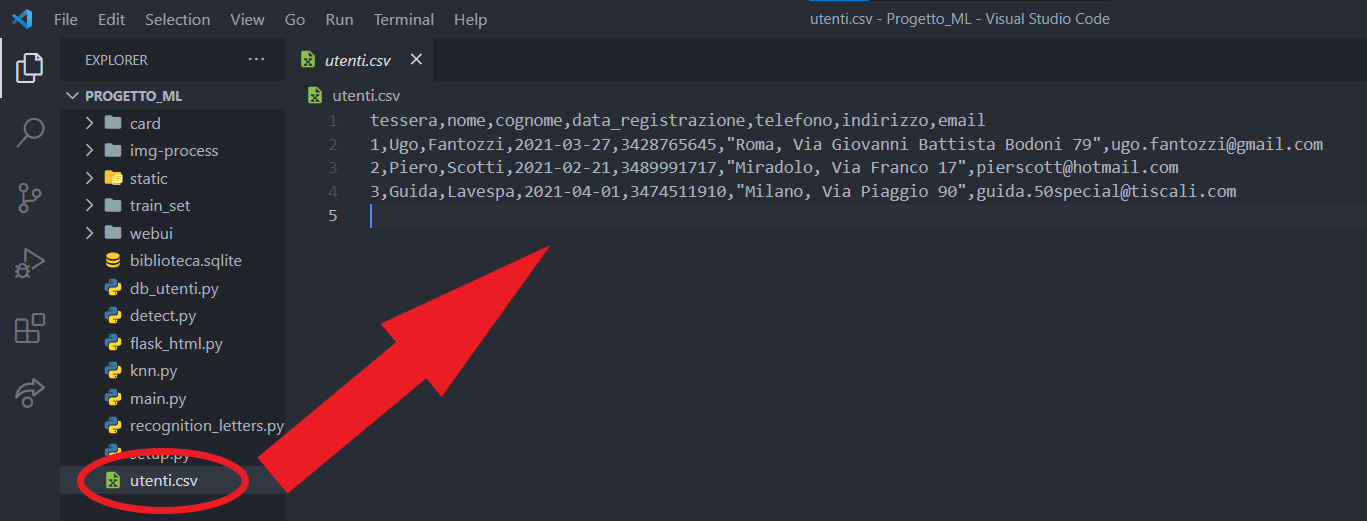
Il nostro progetto prevede la creazione di un sistema di registrazione utente ad un database di una biblioteca. Tramite OCR da noi implementato si estrarranno in modo automatico nome e cognome dell’utente dal documento d’identità, pre-inserendoli quindi nel form di registrazione al database.

## **Setup**

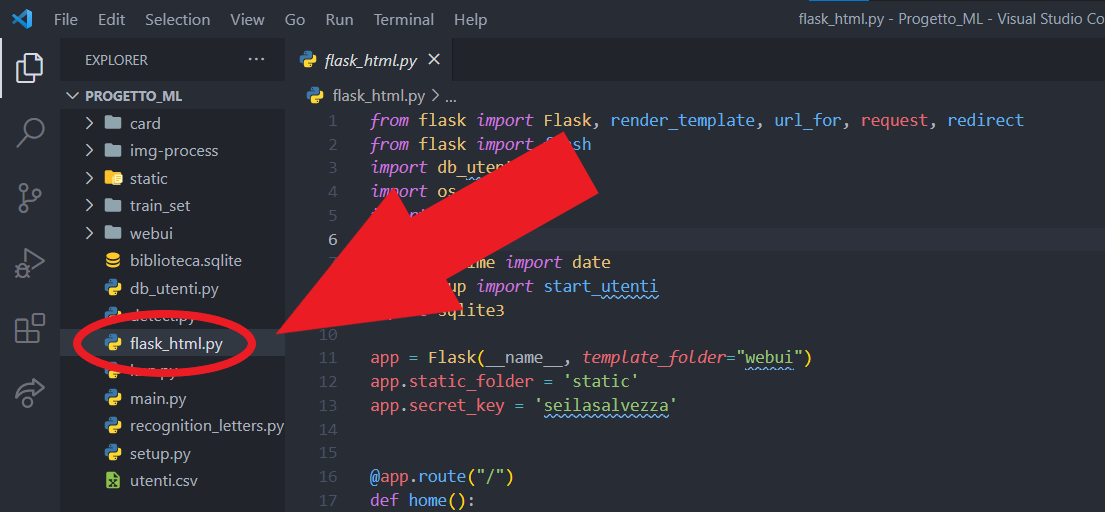
Il primo utilizzo del programma prevede l’esecuzione dello script setup.py:



Lo script creerà quindi un database Utenti in SQLite3 e un file csv, entrambi popolati con i dati di utenti da noi forniti. Database e csv si aggiorneranno poi con i dati trasmessi tramite la nostra implementazione web.



Per avviare la pagina in HTML e quindi procedere all’inserimento di nuovi utenti nel database è necessario eseguire flask\_html.pye accedere alla pagina che verrà mostrata nel terminale.

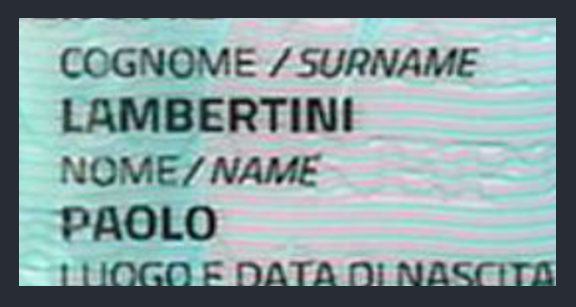
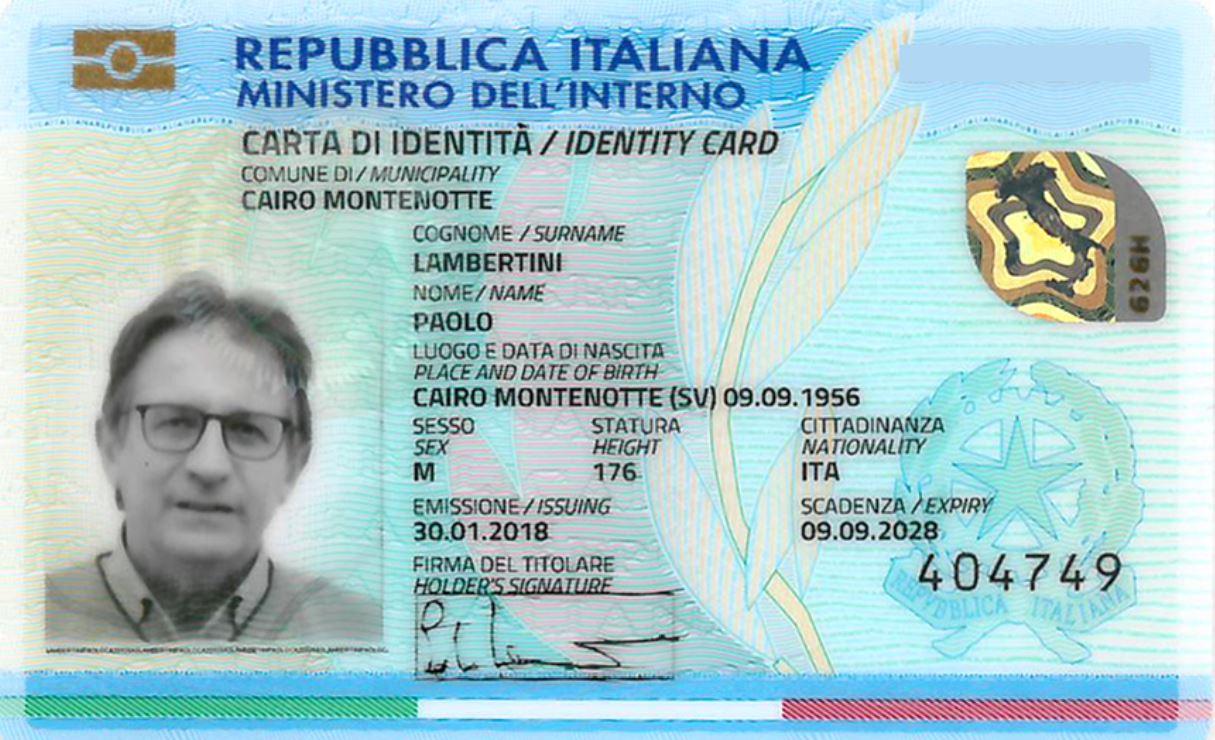


## 

## **OCR & KNN**

L’immagine caricata nella nostra homepage HTML verrà quindi salvata nella cartella *card* e processata per estrarre nome e cognome dell’utente che verrà aggiunto al database.

Per prima cosa l’immagine verrà ritagliata attraverso la funzione crop(). Come risultato si otterrà un’immagine che conterrà la sezione NOME/NAME e COGNOME/SURNAME della cartà d’identità processata.



L’immagine ritagliata passerà quindi attraverso la funzione transform\_image()

che tramite il thresholding e boundingrect isola e ritaglia il nome e il cognome



Dopo aver isolato il nome e il cognome utilizzeremo le funzioni grayscale()*, binary()* e dilation() per processare l’immagine e renderla più pulita, la funzione countours()per scontornare le lettere, e infine rect()per ritagliare e salvare le lettere di nome e cognome in cartelle apposite.



Abbiamo utilizzato il KNN come algoritmo di riconoscimento delle lettere.

Come prima cosa abbiamo scaricato font da utilizzare come train e test set. Li abbiamo quindi importati attraverso ciclo *for* come numpy arrays, assegnando a ciascuna lettera del singolo font una label. Infine abbiamo utilizzato train\_test\_split() per dividere il tutto in 80% di train set e 20% di test set. Con la funzione di sklearn predict() facciamo quindi passare tutte le precedenti lettere ritagliate come argomento e otteniamo la previsione della lettera.

## **Flask**

Tramite Flask abbiamo creato la route per le pagine HTML.

Per verificare se l’utente fosse già registrato nel nostro database abbiamo implementato una funzione che verifica se il numero di telefono inserito nel form HTML sia già presente nel database (ci siamo serviti della funzione *fetchall* per recuperare i dati dal database SQLite su cui effettuare il check); abbiamo eseguito lo stesso controllo per il campo email. Ottenuti i dati tramite *request* dalla nostra pagina HTML di registrazione, abbiamo quindi passato i dati ad un dizionario *Utente* e, tramite la nostra funzione sql\_Utenti(), caricato i dati nel database SQLite.

## **Implementazione Web**

Abbiamo creato tre pagine HTML diverse:

* index.html: pagina iniziale dove si carica l’immagine di una carta d’identità;
* registrazione.html: pagina dove si verrà reindirizzati dopo aver eseguito l’upload della carta (i campi nome e cognome verranno compilati automaticamente ma avranno possibilità di modifica; oltre a questi campi vi sono quelli per il recapito telefonico, l’indirizzo di residenza e quello per l’email);
* reg\_effettuata.html: pagina di conferma di avvenuta registrazione con reindirizzamento alla pagina *index.html*.

## **Problemi riscontrati**

Il problema principale riscontrato è lo scarso numero di carte d’identità ottenibili su internet e la loro scarsa qualità. Questo non ci ha permesso di testare in modo efficace il nostro modello di KNN e OCR in quanto la scarsa qualità delle immagini reperite rendeva difficile il riconoscimento delle lettere tramite il confronto tra arrays di pixel. Per questo abbiamo deciso di utilizzare il thresholding per il riconoscimento delle lettere piuttosto che l’immagine originale o in scala di grigi, garantendo così una qualità migliore delle lettere. Abbiamo avuto difficoltà anche nel reperire un dataset di font da utilizzare per il training e testare il nostro modello (il nostro dataset finale presenta “solo” 11 sub-dataset formati dalle 26 lettere dell’alfabeto in maiuscolo).

## **Conclusione**

Tramite l’utilizzo di un database SQLite3, HTML e il micro-framework di Python Flask, abbiamo implementato un sistema in grado di estrapolare ed elaborare dati da un documento di identità, per effettuare quindi il riconoscimento dei caratteri attraverso un custom OCR basato sull’algoritmo di machine learning KNN e salvare i dati in un apposito database.

Reputiamo che il nostro modello di OCR/KNN sia efficiente e ben costruito. Tuttavia la scarsa disponibilità di documenti d’identità o di dataset in internet rende difficile un approfondita sessione di test del nostro algoritmo.